Research Summary: Coding a DeFi arbitrage bot

summary defi dex arbitrage-bots

tina1998612 Research Team Apr 6

> TLDR:
>
> - The author provides a detailed guide to coding a DeFi arbitrage bot. The bot uses flash loans to borrow assets from dYdX and sells them on 1inch exchange when profitable.

Citation

- Extropy.IO . Coding a DeFi Arbitrage Bot, Medium. Oct, 29, 2020. Accessed on: Mar, 16, 2021. [Online] Available: Part 1: Coding a DeFi Arbitrage Bot | by Extropy.IO | Medium , Part 2: Coding a DeFi Arbitrage Bot — part 2 | by Extropy.IO | Medium

Link

- Part 1: Coding a DeFi Arbitrage Bot | by Extropy.IO | Medium
- Part 2: Coding a DeFi Arbitrage Bot — part 2 | by Extropy.IO | Medium

Core Research Question

- How can an arbitrage between DEXes be automatically performed using flash loans?

Background

- **Arbitrage** is the purchase and sale of an asset in order to profit from a difference in the asset's price between marketplaces.
- **Price slippage** refers to the difference between the expected price of a trade and the price at which the trade is executed. It usually happens when the market is highly volatile within a short period of time or when the current trade volume exceeds the existing bid/ask spread.
- **Flash Loan** is a type of uncollateralized loan that is only valid within a single transaction. It can be implemented through a smart contract. The transaction will be reverted if the execution result is not as expected.
  - For more details on flash loans, please refer to the research summary "Attacking the DeFi Ecosystem with Flash Loans for Fun and Profit ".
- **Decentralized exchanges (DEX)** are a type of cryptocurrency exchange which allow peer–to–peer cryptocurrency exchanges to take place securely online, without needing an intermediary.
- **DEX aggregators** source liquidity from different DEXs and thus offer users better :han any single DEX.

Skip to main content

- **Liquidity pool** is a collection of funds locked in a smart contract to provide liquidity for DEXes. The advantage of a liquidity pool is that it doesn't require matching orders between buyers and sellers, and instead leverages a pre-funded liquidity pool with low slippage.
- An **Orderbook** consists of a collection of bid-and-ask orders. Orders are matched and executed only when a bid and ask price are the same.
- An **Automated Market Maker (AMM)** uses a liquidity pool instead of an orderbook and relies on mathematical formulas to price assets. The assets can be automatically swapped against the pool's latest price, making it more efficient than traditional orderbooks.
- **Wrapped ETH (WETH)** is the ERC20 tradable version of Ethereum. WETH is easier to trade within smart contracts than ETH is. Users can also revoke access to their WETH after sending it to an exchange, which is not possible with ETH.

Summary

- The author describes the advantages of DeFi arbitrage over centralized exchanges:
  - DeFi
    - Insolvency risks are minimized as smart contracts execute automatically following predetermined parameters. A trade will be reverted if it cannot be executed as expected.
    - Traders can perform arbitrage using borrowed funds with flash loans.
  - Centralized exchanges
    - Since a trader cannot execute trades simultaneously, they may suffer from price slippage if a trade is delayed.
    - Traders need to own funds or borrow them from a bank.
- Arbitrage between DEXes that use AMM
  - Popular platforms
    - Kyber Network , Uniswap , Balancer , and Curve Finance .
  - Result
    - Bring prices into efficiency between two liquidity pools
  - Scenario
    - When the pools on different DEXes offer different prices to exchange assets.
  - Execution
    - Exchange from asset A to asset B on one pool and exchange it back on another pool to benefit from the price spread between two pools.
- Arbitrage between DEXes that use classic orderbook
  - Popular platforms
    - Radar Relay , powered by the 0x protocol .
  - Scenario
    - Traders can fill limit orders from a DEX and then see if the tokens acquired could be sold to any other liquidity pools for better returns.
- The author describes the basic operation of an arbitrage bot.
  - For example, to arbitrage the pair WETH/DAI:
  - The bot will query the 0x API looking for WETH/DAI pair limit orders
    - The 0x API can get the limit orders for a currency pair from every exchange that uses the 0x protocol.
    - Example API url for orders buying WETH with DAI:

Skip to main content

x.org/sra/v3/orders?

page=1&perPage=1000&makerAssetProxyId=0xf47261b0&takerAssetProxyId=0xf47261b0&makerAssetAddress=0xc02aaa39b223fe8d0a0e5c4f27ead9083c756cc2&takerAssetAddress=0x6b175474e89094c44da98b954eedeac495271d0f

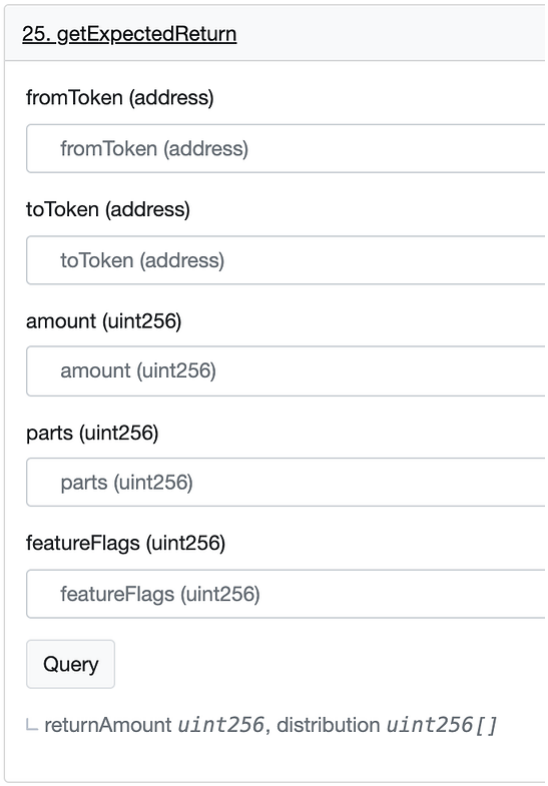
- Parameters
  - the maker token's contract address (WETH)
  - the taker token's contract address (DAI)
- Sample response from the above request


```
- order: {
      signature:
      "0x1bc806e51c5b321ab601a88eb58f5c6b6a5182500aece5408eb189a513ab6c727a6e3afca6cc405b3d8849ad25945
      senderAddress: "0x0000000000000000000000000000000000000000",
      makerAddress: "0x70b6cadaa0fa1e33ae668fb8fcd827be807b489a",
      takerAddress: "0x0000000000000000000000000000000000000000",
      makerFee: "0",
      takerFee: "0",
      makerAssetAmount: "5000000000000000000",
      takerAssetAmount: "710000000000000000000",
      makerAssetData: "0xf47261b0000000000000000000000000c02aaa39b223fe8d0a0e5c4f27ead9083c756cc2",
      takerAssetData: "0xf47261b0000000000000000000000000006b175474e89094c44da98b954eedeac495271d0f",
      salt: "1611201275",
      exchangeAddress: "0x61935cbdd02287b511119ddb11aeb42f1593b7ef",
      feeRecipientAddress: "0x68a17b587caf4f9329f0e372e3a78d23a46de6b5",
      expirationTimeSeconds: "1642305274",
      makerFeeAssetData: "0x",
      chainId: 1,
      takerFeeAssetData: "0x"
  },
```


- The DAI price of 1 ETH = takerAssetAmount / makerAssetAmount

- The bot will then query the 1inch exchange DEX aggregator to determine if there is any open order that could be sold for a higher price on any other liquidity pool.
  - Reason for using 1inch
    - It offers the best rates by discovering the most efficient swapping routes across all leading DEXes.
  - Goal
    - Get the DAI price for 1 WETH.
  - Detailed steps
    - Execute the getExpectedReturn function of the 1inch smart contract function on Etherscan .
    - Parameter
      - Input
        - fromToken: WETH contract address
        - toToken: DAI contract address
        - amount: the number of WETH we are selling, followed by 18 decimal places.
      - Output
        - returnAmount: this number divided by the input "amount" parameter is the DAI price for 1 WETH.
    - Arbitrage may take place if this amount is greater than the one returned by 0x.

Skip to main content



- However, the above opportunity may not actually exist in practice, because we would need both limit orders to exist at the same time for the arbitrage to work.
- The author thus proposes to perform the arbitrage using a flash loan.
  - Steps
    - Get a flash loan DAI from DyDx exchange
    - Buy WETH from 0x using the DAI borrowed with the flash loan
    - Use 1inch to find the best exchange to sell the WETH acquired in step 2
    - Pay back the flash loan DAI and keep the remainder as profit
- The author provides the code and explains how the arbitrage smart contract works
  - The contract inherits the DyDxFloashLoan smart contract.
  - Functions
    - Swap
      - Perform the trade
    - getExpectedReturn
      - Get the current asset price
    - getWeth
      - Turn any ETH sent into WETH
    - approveWeth
      - Approve the 0x proxy to collect WETH as the fee for using 0x.
    - getFlashloan
      - Called whenever a profitable arbitrage is found by the client.
      - All of the parameters for this function will be constructed and passed in from the client script.
    - callFunction
      - Has to be deployed in our smart contract to receive a flash loan from dYdX.
    - _arb
      - Arbitrage function that is called when the loan is successful.
      - Tracks the balance of our smart contract before and after the trade. balance is not greater than the start balance, the operation will revert.

Skip to main content

Method

- The prerequisite for the arbitrage bot is to have a browser with the Metamask extension installed.
- The programming language used is NodeJS.
- Project structure: only two files
  - index.js
    - a node.js server that continuously fetches crypto prices on exchanges looking for arbitrage opportunities, trying to guarantee that the trade is possible before even attempting to execute it.
  - TradingBot.sol
    - a smart contract that gets called by the node app only when a profitable arbitrage is found, it will borrow funds with a flash loan and execute trades on DEXes.
- Detailed setup
  - Install Metamask browser extension
  - Create an Ethereum Mainnet account with some ETH for paying gas fees.
    - Don't use your personal account for this, create a new account for the bot in order to limit accidental losses.
  - Go to Remix online IDE and paste the smart contract solidity code
  - Compile the code using compiler version 0.5.17.
  - Deploy with an initial 100 wei, which is enough for 100 flash loans on dYdX.
- Environment setup
  - By cloning the project's code repository, users will find a file called .env.example inside the /src folder
  - Fill in the fields:
    - RPC_URL : the public address of an Ethereum node, the easiest one to set up is the Infura RPC provider, register an account in order to get an API key.
    - ADDRESS and PRIVATE_KEY: fill in the public Ethereum address of the bot account, and its corresponding private key.
    - CONTRACT_ADDRESS : paste in the smart contract's address that was returned from Remix after the deployment step.
    - GAS_LIMIT : how much gas the contract is allowed to use, leave as 3000000 or decrease to 2000000 which should be fine
    - GAS_PRICE : change this depending on how fast you want the transaction to be mined, see https://ethgasstation.info/ for info.
    - ESTIMATED_GAS : leave as is
- Running the bot
  - Execute the command from the project's root directory.
    - node src/index.js

Results

- The full working code can be found at GitHub – ExtropyIO/defi-bot: Tutorial for building DeFi arbitrage bots .

Discussion & Key Takeaways

Skip to main content] an open-source DeFi arbitrage bot that uses flash loans to borrow assets from dYdX and sells them on 1inch exchange when profitable.

- The author explains the main components of the arbitrage bot and the underlying logic of how arbitrage works.
- After following this tutorial, users can create a working example of a customizable flash loan arbitrage bot.
- The most efficient way to perform a flash loan arbitrage is to continuously fetch the real time prices using NodeJS client and execute the contract with profitable parameters when an opportunity is found.

Implications & Follow-ups

- Arbitrage is a zero-sum game. There are a finite number of arbitrage opportunities for a large group of people competing to find and execute them.
- To make the bot more efficient, the author suggests the following improvements:
  - Consider taker fees when calculating profits
  - Use Partial fills
  - Check orders again
  - Handle failures to continue execution
  - Execute multiple orders simultaneously
  - Dynamically calculate gas fees
- If such arbitrage bot becomes prevalent, the price differences between different DEXes will be minimized. DEX aggregators such as 1inch may no longer be needed as the price differences become more and more negligible in the future.
- It may be interesting to measure the actual APR of running this bot, considering the cost of server hosting and contract deployment.

Applicability

- Interested readers can refer to the working code to have their own arbitrage bot: GitHub – ExtropyIO/defi-bot: Tutorial for building DeFi arbitrage bots .
- Currently, the example only supports flash loans from dYdX. Users can add other flash loan provider's support.

🔗 Research Summary: Attacking the DeFi Ecosystem with Flash Loans for Fun and Profit

🔗 SCRF Terms Glossary and Content Tags

Skip to main content

Barry Research Team **Apr 7**

Has anyone published the results of running the bot? Even if it's consistently not profitable it would be useful to understand why.

In Flash Boys they setup multiple globally distributed nodes in order minimize the impact of latency a node can experience. Any consideration given to latency?

Given that price movements create arbitrage opportunities are there any studies examining the relationship between price volatility and gas prices?

jmcgirk Research Team **Apr 9**

Can you give us a sense of how important these bots are to the DeFi ecosystem? Are they simply a nuance or do they provide some kind of stability to the system? I'd also be really curious about how arbitrage bots and trading bots fit into the debate about Miner Extractible Value.

cipherix Research Team **Apr 9**

Most DeFi applications rely on arbitrageurs to function properly so these trading bots are critical for markets to reach price efficiency.

When it comes to node latency, that seems to be of less relevance in recent times because of new approaches to MEV. Don't get me wrong, it's still important to have sufficient data on the different corners of the network's topology. Since the mempool functions as an auction, you need to know if another arbitrageur outbid your tx and whether it's worth it to engage in a bidding war.

However, it has been observed that new approaches to MEV entail sending transactions directly to the mining pool and issuing payments out–of–band (without even sending it to the mempool). That circumvents the need for low latency as miners will receive your transactions directly, thereby circumventing the need to engage in bidding wars.

jmcgirk Research Team **Apr 15**

What other non–human (or quasi–) human actors are lurking in the DeFi space, @tina1998612 ? Seems like a forbidding environment for a retail investor to enter. Are there any other takeaways from Flash Boys and Towards Understanding Flash Loan and Its Applications in DeFi Ecosystem that you think might be pertinent to this discussion?

Skip to main content

@Larry_Bates had some really interesting thoughts about anonymity and identity, and who might be lurking within the dark forest…

@Barry where do you think all of this ends up in five or ten years?

Barry Research Team **Apr 16**

> jmcgirk:
>
> where do you think all of this ends up in five or ten years?

Hopefully some of the research and development today around scalability will be online in the next five years to increase the transactional capacity of the network(s). Arbitrageurs can then continue to ensure prices are properly reflected across markets with less impact on cost of transaction execution globally.

---

jmcgirk Research Team **Apr 20**

This arbitrage bot works specifically for 0x and 1inch exchanges. How much tweaking would something like this need for a different exchange? Or what about a non-Eth-based DEX? I wonder what will happen now that the code is out in the wild, I guess in theory it'll make arbitrage unprofitable on those DEXes. I wonder how all of this parallels with the introduction of high frequency trading in TradFi.

Sean1992076 **Apr 21**

I'm also interested in Miner Extractable Value to include this framework. DeFi Arbitrage bot is a very funny thing to implement, but it does not guarantee profitable exclude some parameters in this method such as Miner Extractable Value, which means your transaction may not be successfully taken by the miner. And the different DEX may have different MEV to calculate.

---

jmcgirk Research Team **Apr 22**

@kanad had a question in our chat: "What is the underlying technical difference between bots and flashbots?" @Larry_Bates you had a great response, but perhaps @tina1998612 you might like to weigh in as well @jyezie

---

Kirsty_G **Jul 1**

We recently wrote an article on the MEV Crisis with some solutions. Here's the link: https://extropy-io.medium.com/illuminating-the-dark-forest-748d915eeaa1

Skip to main content

jmcgirk Research Team **Jul 1**

@Kirsty_G – thank you so much for the update, looking forward to reading it!
@tina1998612 – I'd be really interested to see what you think of this.

---

zigomi **Sep 18**

Will this

> Coding a DeFi Arbitrage Bot | by Extropy.IO | Medium , Part 2: Coding a DeFi Arbitrage Bot — part 2 | by Extropy.IO | Medium

work as is, given that there're front-run bots out there? Or will it necessitate some tweaks first?

Laurence_Kirk **29d**

Yes it requires tweaks, we did find that it was being front-run

---

zigomi **29d**

What would be a possible fix? Would running it via Flashbots fix it?